Lineate

The New View: How Data-Rich Dashboards Have Changed The Game

Addressing complexity in the data-rich dashboards of today.

CONTENTS

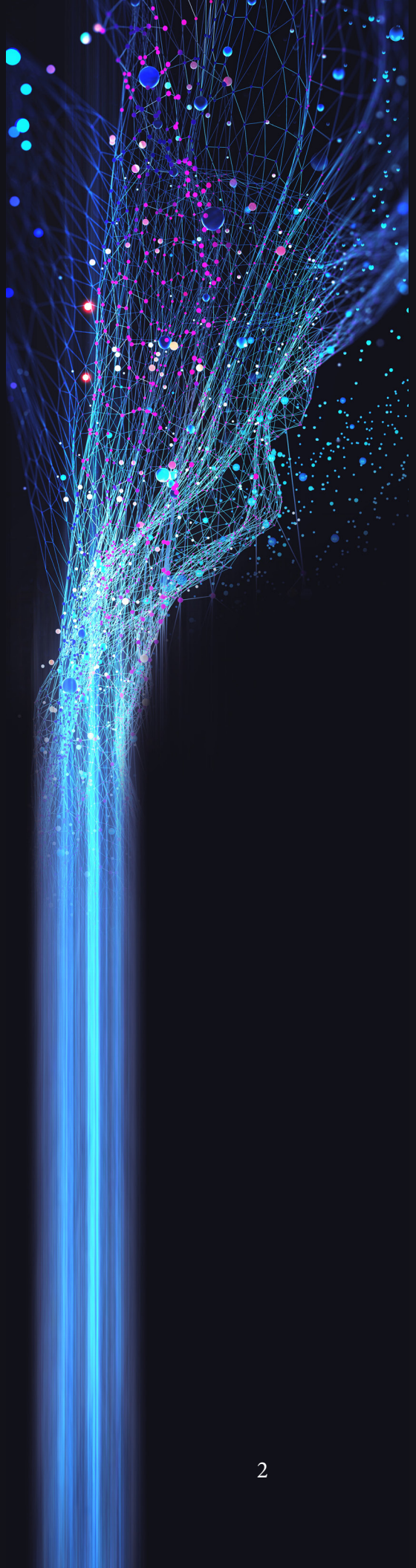
Introduction

The amount of data collected by organizations is exploding, growing at an expected 26% compound annual growth rate between 2015 and 2025. The emerging influx of data comprises event-level information from various applications, user devices, sensors, and low-level operational processes. It has the potential to open doors to new ways to grow business, ranging from personalization, predictive optimization, tailored product recommendations, and countless opportunities for direct relationships with customers.

A crucial step towards this is to aggregate and visualize the data in ways that assist in developing these processes, be they data science hypotheses, optimization paths, or product ideas. But such aggregation and visualization is difficult with large amounts of semi-structured data.

It requires that we prepare the data such that we can explore it almost instantaneously, and that we preplan the interactions from both user experience and data optimization perspectives.

What is a data-rich dashboard?



A dashboard is distinct from a report in that it provides multiple views into the data and allows a user to explore it interactively. Traditionally, dashboards were a convenient set of high-level metrics organized to conveniently give groups of users what they need at a glance. The data-rich dashboard takes this same idea and incorporates into it low-level, semi-structured events. These events were often not captured at all in the past, or at best been dumped into a data lake for occasional ad-hoc queries, or summarized as high-level aggregates and exposed as KPIs. A data-rich dashboard exposes these low-level events at scale. It takes granular data points as they flow in, and provides an immersive and extremely responsive way to make sense of them.

Under the hood, the main difference driving this evolution is the large number of events converging into a data stream. These events range from usage logs around web or mobile applications, to feeds coming in from the ever increasing array of devices on the internet of things, to incredibly low level operational data that is increasingly being collected for systems such as those in creative production, automation, or marketing. These events pose two challenges beyond those in traditional dashboards. First of all, there are a lot of them. Sometimes on the order of billions per day. And secondly, they are often not very structured, making queries of them relatively specialized and not conducive to real-time interaction. The primary challenges of a data-rich dashboard is to make it feel natural to explore such data, and to be able to respond and drill down on such data essentially immediately.

Kinds of dashboards

Dashboards are often categorized into three types:

1 Strategic Dashboards

Infrequent updates to provide high-level information to executives

For example: company or departmental scorecard, KPI progress against targets, project ROI or financial reports

2 Operational Dashboards

Allow employees “in the trenches” to get feedback on their operational activities and react to them

For example: ad campaign performance, user engagement with various content, team productivity or cost metrics

3 Analytic Dashboards

Explore data to understand what is happening behind the scenes, empower ideation and build hypotheses

For example: Cross-cutting filtering of sales, marketing attribution

When we talk about making dashboards data-rich, we're focusing on the latter two. Both operational and analytic use cases offer the potential for people to quickly and seamlessly explore data down to extremely granular events.

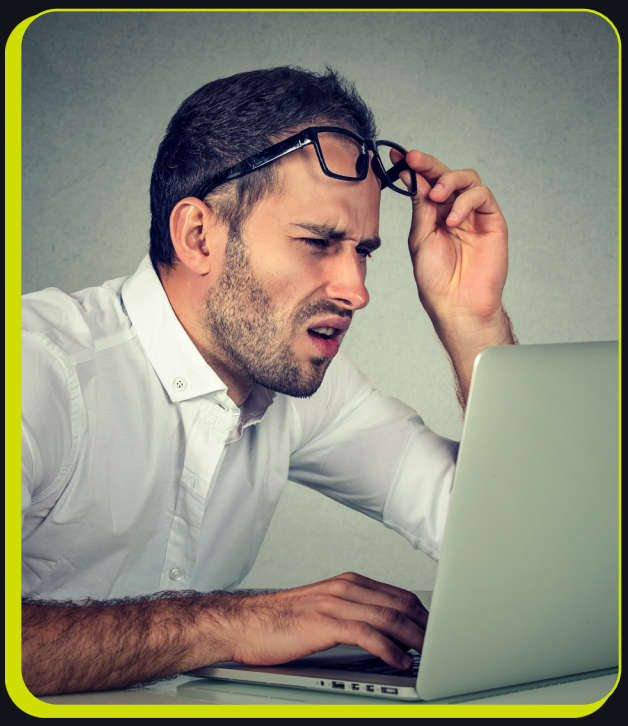
For operations, this can mean providing point-in-time and near real-time snapshots of operations to adjust or optimize parameters on the fly (e.g. retargeting an advertising campaign to where it is performing best.

For analytics, it can mean providing visualization tools for your data scientists to assist in hypothesis generation, or it can provide the rest of users and understanding of **why** algorithms are doing what they are, and to help them make sense of the flood of aggregated information.

Data-rich dashboards

	OPERATIONAL	ANALYTIC
ACTIVITIES ENABLED	Enable taking action in ongoing activities, showing up-to-date information in near realtime.	Enable ideation by exploring and visualizing data in fluid ways.
EXAMPLES	Breaking down ad campaign performance along different metrics so the campaign can be optimized on the fly.	Give data scientists a high-level understanding and way to see how data interrelates intuitively, to guide ideas for deeper data mining to perform.
	Present real-time insight into content virality to kick off the creation of similar content or to capitalize on engagement.	Give non-technical users a sense of how algorithms are making their decisions, to build trust.
CHALLENGES	How to aggregate large volumes of events coming in quickly?	How to make the dashboard instantaneously responsive so users can traverse and drill down into very granular data wihout waiting?
	How to query and load this data so front-line systems and dashboard performance is not impacted?	How to make it transparent which data correlates with which, so users can explore intuitively?

The twin concerns of exploring data in real time



For both Operational and Analytic dashboards, it is imperative that users be able to interact with the dashboard and get an immediate response. Load delays or clunky interactions will quickly render using the system a chore, defeating the purpose of making it easy to identify trends to inform decisions. The central challenge in building them involves how low-level event data can be presented intuitively and quickly. We do this through preplanning how data is going to be accessed, and optimizing data retrieval paths along those dimensions. Before we get into how we do that, we'll state the two rules of thumb we would want for any data-rich dashboard.

2 ‘Rules of Thumb’

#1

User interactions should have immediate results.

Every action a user takes should result in the dashboard fully updated within 500 milliseconds. Since any attempt to drill down or query change can result in reloading multiple components across the dashboard, these queries need to be able to run in parallel and complete within a specific time bound. The event data will not naturally fit to within these constraints, so we must preprocess and preaggregate so that we can succeed.

#2

Exploration should be natural and intuitive.

The dashboard must be optimized and designed around the activities that users expect to occur without interruption. For example, when rendering geospatial data, we assume the user will start scrolling through the maps, and the surrounding regions must be ready to be displayed as she does so. For event data, we expect the user to refine the time window she is looking at and will only know the right granularity when it appears. Since it is impossible to resolve completely arbitrary activities in near real-time, we need to develop against the use cases that matter.

In short, we must have the right data, in the right place, ready for the user’s next move.

Challenges of querying event-level data

As we incorporate granular data into the dashboards, the data challenges around these become considerable. An old-school dashboard might report on the level of individual sales or conversions, which would probably involve handling up to somewhere on the order of thousands of records per day. When we start to include lower-level user events (individual device actions, operational logs, ad campaign impression-level data, etc.), it's not unusual to be dealing with millions or even billions per day. Even when the amount is lower, this kind of data is typically less structured than traditional operational data, and often not conducive to querying. Since we gave ourselves a mandate of responding immediately, we'll need to carefully design our data architecture to make this a possibility.

> "It is impossible to create a purely general purpose system that says "let me explore all my granular data, super fast, and query by any dimension we want."

There are powerful database products like Snowflake or Vertica which are designed specifically to handle analytical queries across large data volumes. These are often crucial components in a data warehouse, and offer the ability to create and run complex queries over large amounts of data.

They are not, however, designed to be able to run large numbers of queries in the half-second latency requirement we set out for ourselves. This isn't because of any flaw in the products — they are general purpose analytical databases which are designed for flexibility. It is simply not possible to create a general-purpose database that can run any kind of query over a large dataset in a fixed amount of time. Fortunately, a real-time dashboard is not attempting to answer arbitrary questions, so there are other ways to solve this problem.

To achieve this, we must design the data interactions in the planning stages. We carefully consider how users are likely to interact with the data and what kinds of questions they will be asking. By designing the dashboard around such use cases, we can achieve enormous improvements in performance and scale through careful preaggregation of data. We then use appropriate databases to create the custom preaggregated indexes we need for the use cases we've designed.

In the case of event data, almost all interactions begin with querying over a single time range. So we start designing our data model through period-based preaggregations of useful partial values, and ensure we have a database that can query across time ranges efficiently (there are time series databases such as Apache Druid and InfluxDB that are often good choices, as are columnar databases such as Clickhouse or Cassandra, depending on the data and use cases.) We know the user is going to select a time range, and we know to preindex the events a single range within this range. We then take those preaggregated events and combine them in ways that depend on the action the user takes.

These aggregates must be kept updated with events as they flow in. The write load is by definition intensive, and performance of the data indexes needed to take that into account. Compounding this is the fact that event data is presumably coming from multiple disconnected systems operating at different network tiers and connecting to to different devices. The nature of distributed systems means that events are unlikely to come in sequentially, so streaming the data into the aggregate indexes needs to be able to rewind and recover without impacting dashboard performance. These are solvable problems, but need to be considered at the time the data architecture is designed.

Querying large amounts of data in parallel is fundamental to a data-rich dashboard. To do so effectively doesn't require that everything be known up front, but it does necessitate that data efficiency is built in to the development process from the start. The data needs to be preaggregated in ways it will be queried at scale and it needs to be streamed into this system in a way that minimizes impact on user-facing systems and the users of the dashboards themselves. Latency targets need to be built into the development process as first class requirements, with code-level timing assertions built in throughout the engineering and testing phases. The return on this is a system that allows users to work with and explore potentially huge pools of data as if it were all immediately available.

Managing complexity using drill paths

We just discussed that we need to properly preaggregate the data if we want to make a dashboard immediately responsively. Done correctly, this enables us to make something immediately responsive. But there is a problem here. The more dimensions we allow the user to explore, the greater the burden of preindexing data becomes.

How do we constrain the kinds of queries users will make enough to render them nearly instantly, while at the same time allowing them to comfortably explore data in different dimensions?

For each dimension we allow the user to explore, we need to maintain the set of preaggregates that allow that data to be returned quickly into the dashboard. In addition to that, since the dashboard allows users to drill down by various attributes as they use the dashboard, it means that we need preindexed data not simply for every dimension we are exposing, but for every combination of attributes they may invoke.

This can result in a “combinatorial explosion” of preprocessing and disk space as the number of supported queries increases. Therefore, we need to control this at the application design level. This is where the concept of “drill paths” comes in.

Definition: A drill path is a set of hierarchical or naturally related objects of the same class ordered in the way they are meant to be explored.

Sample Drilldowns

1

Domain: Publishing

High level Metrics

- Time on site
- Page views
- Conversions

Drilldowns

Cohort → individual → page view → action

2

Domain: Advertising

High level Metrics

- Conversion rate
- Cost per click

Drilldowns

Campaign → segment → user → conversion path → impression

3

Autonomous Driving

High level Metrics

- Accident rate
- Fuel consumption

Drilldowns

Region→ road → individual → trip → state change

By defining a supported set of drill paths, we create a sandbox that allows us to manage the number of dimensions we preaggregate against. We can then design a preaggregating data indexing layer that is able to present this data with sub-second latency, allowing the dashboard to respond immediately to the set of interactions we allow the user to drill down into.

Guiding the user with contextual design cues

Drill paths provide the additional benefit of making the dashboard intuitive to use. With a data-rich dashboard, a large number of semi-structured events are explored in different ways, which generally requires a significant number of visual components that interact with each other. This can get overwhelming fast. We can leverage the same drill paths we define to provide visual cues of which components are conceptually linked to each other, and provide a natural way to drill down to explore the data in final detail.

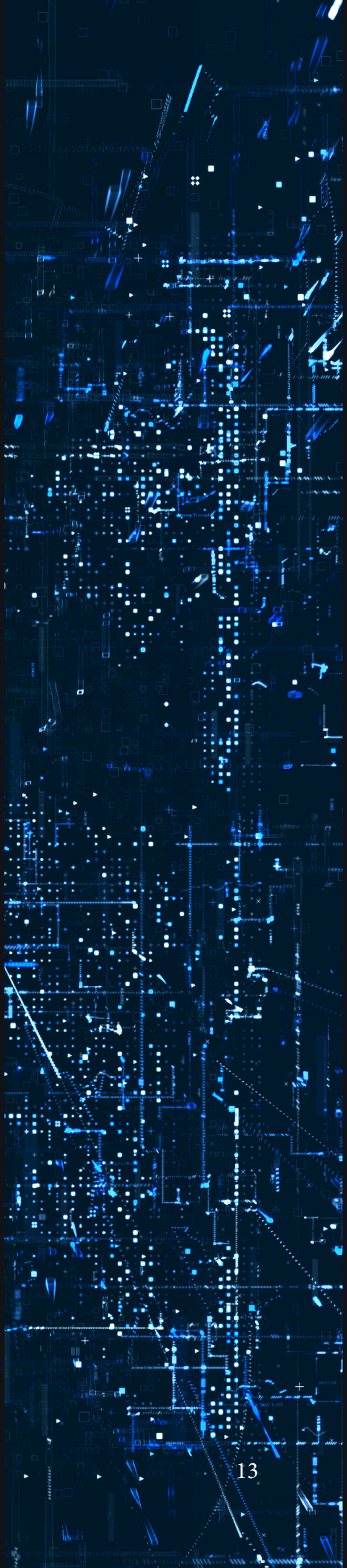
Every dashboard consists of a series of charts, tables, indicators, and controls that illustrate various properties of the entity being looked at. As a user interacts with one of these widgets, the other widgets dynamically adapt to provide further insight into the data (this is what makes it a dashboard and not simply a report.) The user probably isn't thinking explicitly in terms of drill paths, so the user experience should focus on minimizing the mental work required to understand which widgets correspond to which.

To illustrate: Imagine a dashboard which contains a pie chart of conversions by region, and a bar chart of conversions by state. These two widgets would be color coded in the same way, to give the user a visual indication that they are related windows into the same drill path. Clicking on a region in the pie chart would cause the bar chart to break down by states in that region. Drilling down further in the pie chart might refresh to show breakdown by state, and simultaneously update the bar chart to show cities within the state. Note that while it would also be possible to click on a region to determine the regional sales by day of week, this would break the natural visual correspondence between these two widgets. We would instead use a separate set of components with a different drill path and color scheme, adding more components but simplifying the interaction.

Optimizing for time and space

In aggregate, series of events tell us a lot, but each isolated low-level event is mostly meaningless. So why allow drilling down to this level? Because it allows users to see the foundational elements on which the metrics are being built, providing context and building trust in the model.

If users are able to see the lowest level data available, they will feel more comfortable extrapolating from it, even if they're not using that data directly on a day to day level. The nature of this data allows for some standard optimizations we can do to streamline the user experience.



Using maps effectively

Maps are, of course, the natural way to represent geospatial data and get intuitive insight from it. If you're showing someone a map, you should expect them to spend significant time drilling up, down, and scrolling around as they look for patterns. Since there is a potentially huge amount of data underlying the maps (at minimum, the considerable data needed to render the map itself), this is a common place for pitfalls in implementing a dashboard.

	NAIVE WAY	BETTER WAY	BEST WAY
IMPLEMENTATION	Load the map regions on demand.	Load a buffer region around the map, to anticipate user actions.	Break into tiles and predictively load the ones most likely to be needed.
PROS & CONS	Simplest to implement (same as most other dashboard widgets.) Slow and choppy user experience	Allows smooth scrolling across the map. Potentially massive data loads. Also breaks down under rapid user movements across large areas.	Extremely powerful and flexible. Significant development effort and planning around drill paths needed.

Use columnar data storage

Since every event takes place at a specific date and time, almost every query across events will be hitting a time period in one way or another. That means that minimizing the work involved in scanning a range of dates is of paramount importance. We have found columnar databases such as Apache Druid or Clickhouse to be good choices. If we are able to cluster time entry records physically close to each other when we write them, it makes scanning across them to resolve queries along various dimensions vastly more efficient than we might otherwise achieve using a standard relational index.

Plan for multi-tenancy

Even though a dashboard may be envisioned as an internally facing tool only, they tend to evolve in unexpected ways, and sometimes need to be published more broadly. A lot of the recommendations we made around drill paths come from the experience of seeing different groups of users interrelating with the data in different ways. By planning for the supported and unsupported dimensions up front, it adds great flexibility downstream. Even if the dashboard is never published, designing as if it might be will likely result in a better thought out visualization.

Model domain data as a graph using GraphQL

Lineate likes to use GraphQL as protocol for transferring data to a user interface. We find it works especially well for dashboards. As dashboards get richer, it becomes less likely that everything desired is represented by a formal schema beneath them. When we define drill paths, we're in effect defining various projections of this semi-structured data. GraphQL provides a nice way of modeling the drill paths accessible to the dashboard. On top of this, GraphQL provides a nice heuristic for only pulling the specific data needed to render a view, which is ideal for rapid development and limiting bandwidth.

Conclusion

The explosion of data over the past handful of years has triggered the need for advanced dashboards that go beyond providing a business intelligence tool to departmental stakeholders. The large amount of semi-structured data associated with events makes building data-rich dashboards and quantitatively different exercise than building traditional dashboards. Done correctly, it's an interactive window in which people explore and thrive.

They key in making it all work is being fully immersive and interactive. Each component needs to update with very low latency, and the relationship between data needs to be intuitive and transparent.

- Have all the data ready to be served, as soon as the user needs it
- Plan for the specific drill paths in which the data will be explored
- Make it clear and intuitive how each widget impacts every other

This kind of advanced, data-rich dashboards need to be thought of less as an afterthought and more as a source of innovation and competitive edge. Large amounts of semi-structured event data make developing them a first-class application development challenge.

THANK YOU

lineate.com/contact-us